



# Ray Tracing

Tim Purcell  
NVIDIA

**GP GPU**

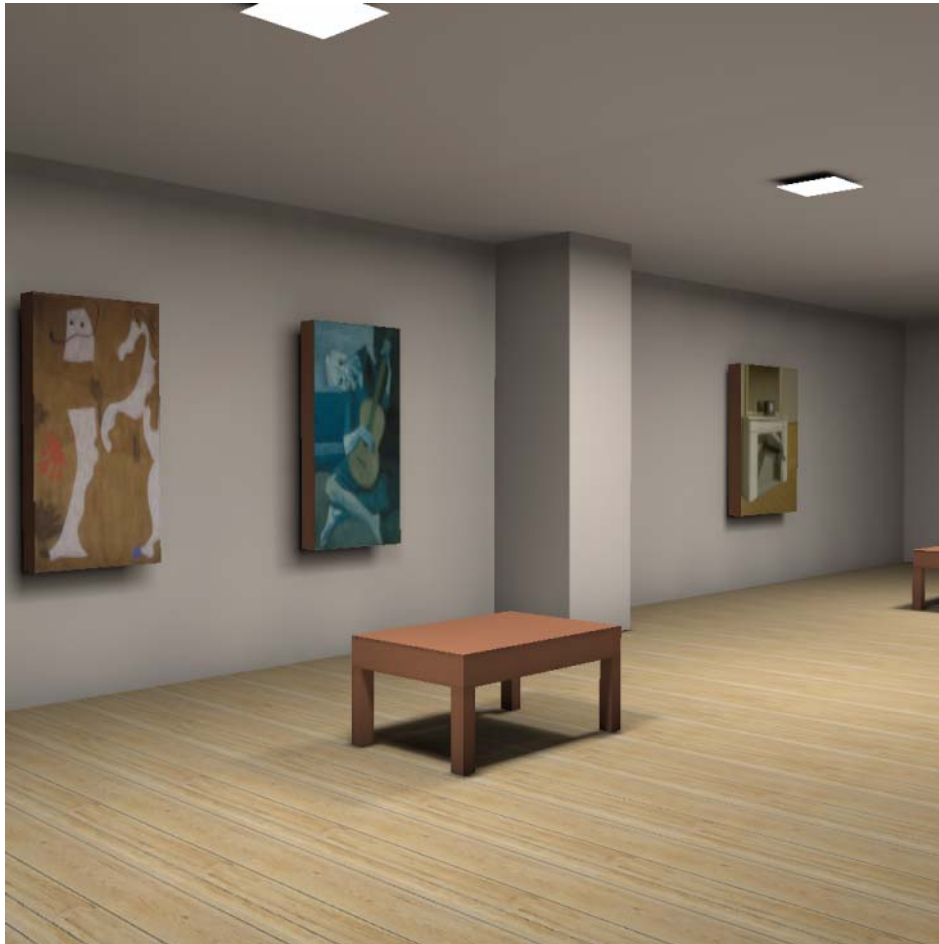
# Small Sampling of GI on GPUs

---

- Much more detail in the included papers
- Lots of other 'global illumination on GPUs' in the literature
  - The Ray Engine [Carr et al. 2002]
  - GPU Algorithms for Radiosity and Subsurface Scattering [Carr et al. 2003]
  - Radiosity on Graphics Hardware [Coombe et al. 2004]
  - Photon Mapping on Graphics Hardware [Purcell et al. 2003]
  - Lots and lots of shadow papers...

# Radiosity

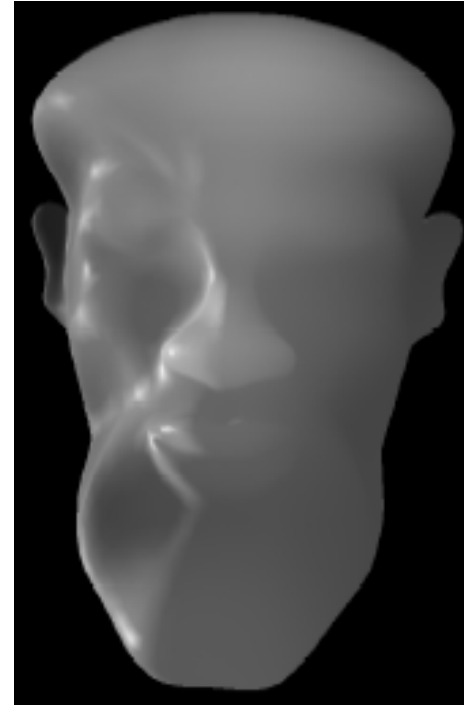
---



Radiosity on  
Graphics Hardware  
[Coombe et al. 2004]

# Subsurface Scattering

---

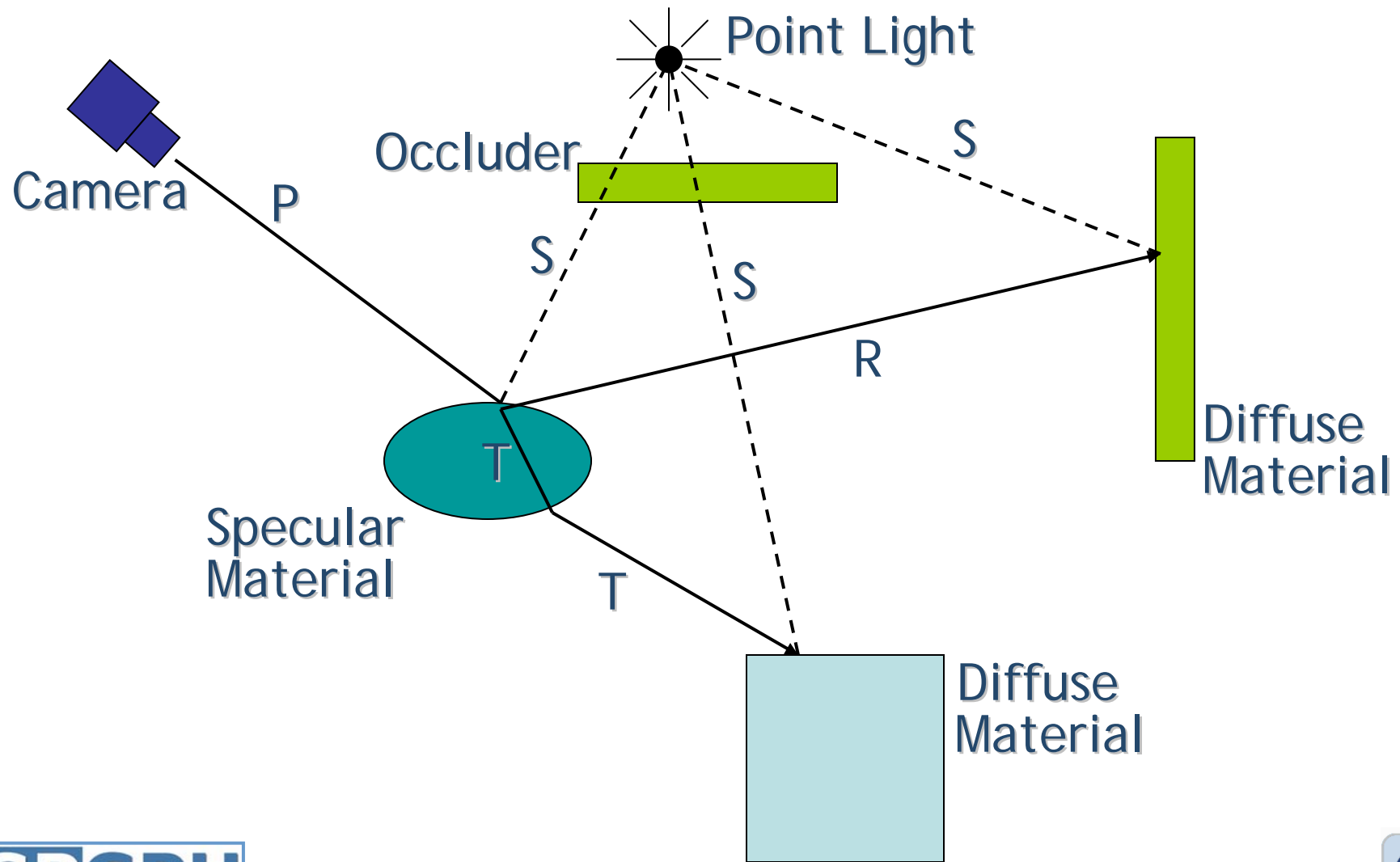


GPU Algorithms for  
Radiosity and Subsurface  
Scattering  
[Carr et al. 2003]

# Ray Tracing

---

# Ray Tracing



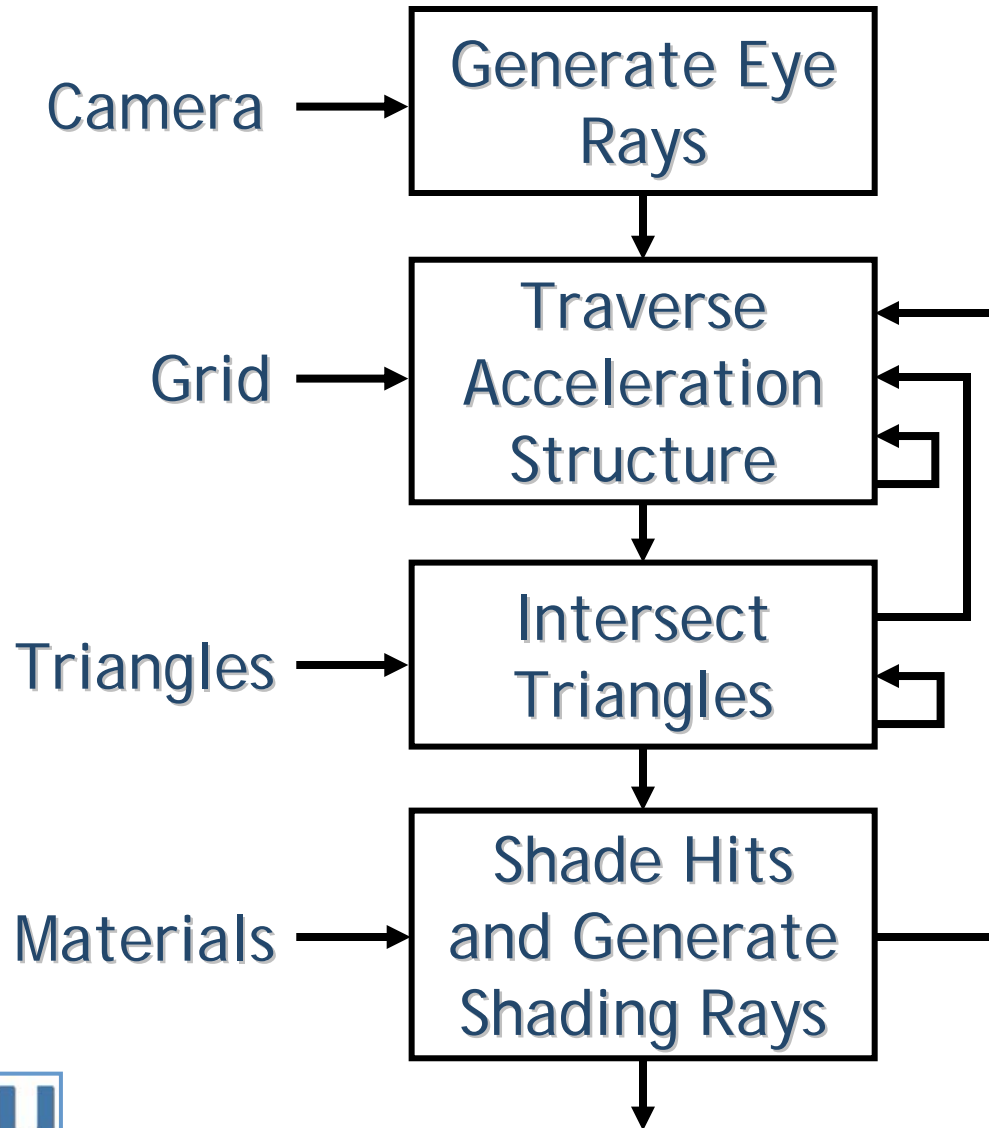
# Implementation Options

---

- GPU as a ray-triangle intersection engine [Carr et al. 2002]
  - Rays and geometry streamed to GPU
  - Intersection calculation results read back
  - Acceleration structure traversal done on host CPU
- GPU as a ray tracing engine [Purcell et al. 2002]
  - Scene geometry and acceleration structure stored on GPU
  - GPU performs ray generation, acceleration structure traversal, intersection, and shading
  - Host provides camera info

# Streaming Ray Tracer

---



# Techniques Used

---

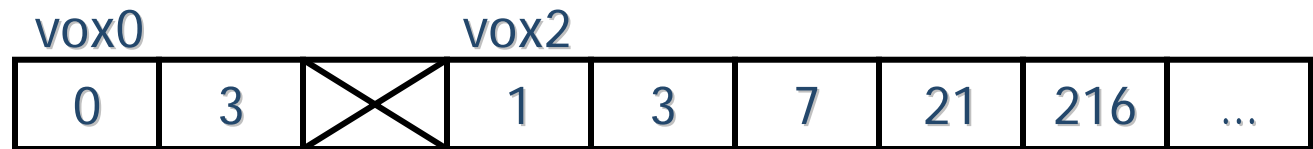
- **Data structure navigation**
  - Texture memory stores data structures
  - Dependent texture fetches walk through data
- **Flow control**
  - Kernel binding based on occlusion query results
  - Efficient selective execution of kernels using early-z occlusion culling
  - Difficulty in flow control disappearing with newest graphics cards
    - PS 3.0

# Texture Memory Organization

Uniform Grid  
3D Luminance  
Texture



Triangle List  
1D Luminance  
Texture

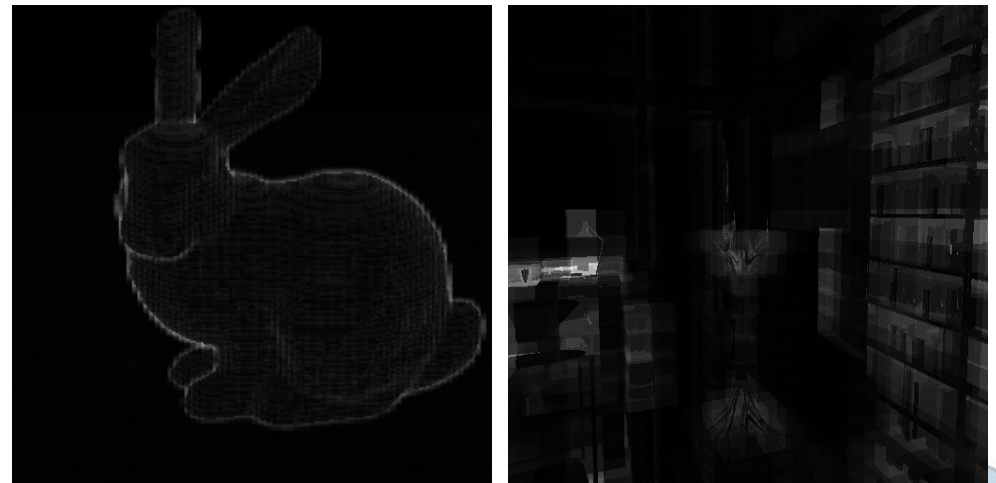
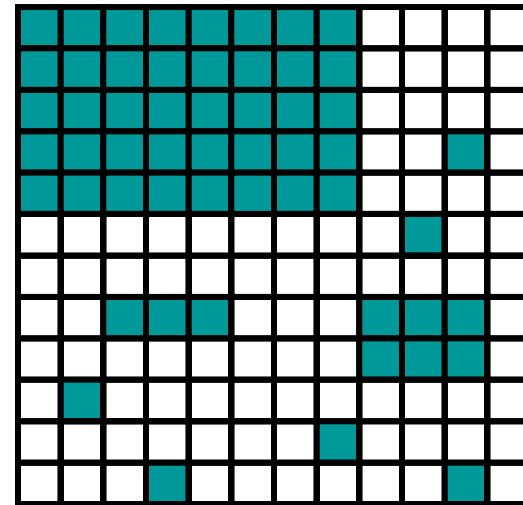


Triangles  
3x 1D RGB  
Textures



# Efficient Selective Execution

- Rendering giant screen filling quad not ideal
  - Not all pixels need to process every rendering pass
- Proposed low-overhead early fragment kill
  - Computation mask
  - Controllable early-Z occlusion culling
- Trade computation for bandwidth



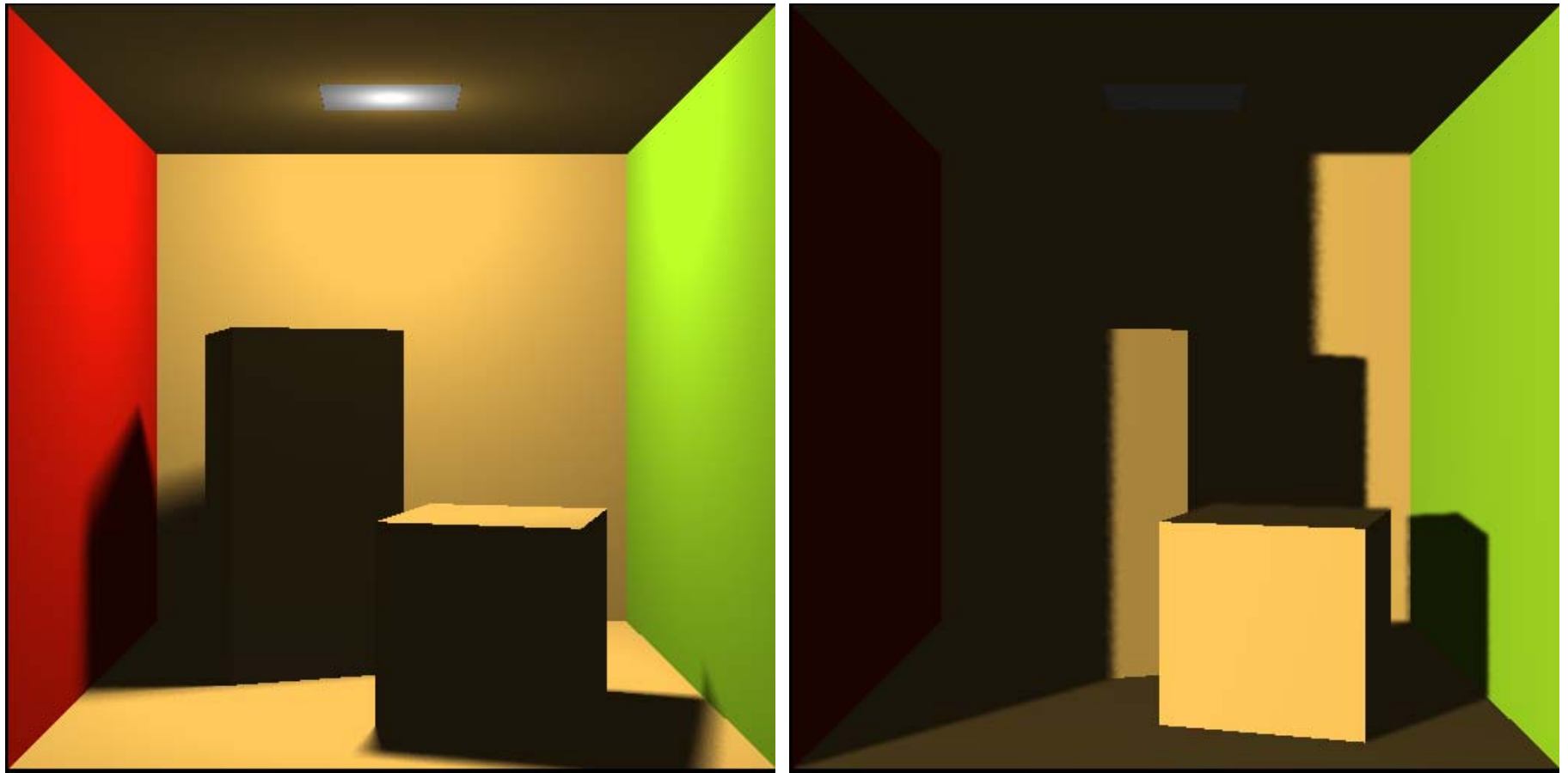
# Original System Implementation

---

- ATI Radeon 9700 Pro (R300)
- ATI Fragment Program

# Cornell Box - Ray Traced Shadows

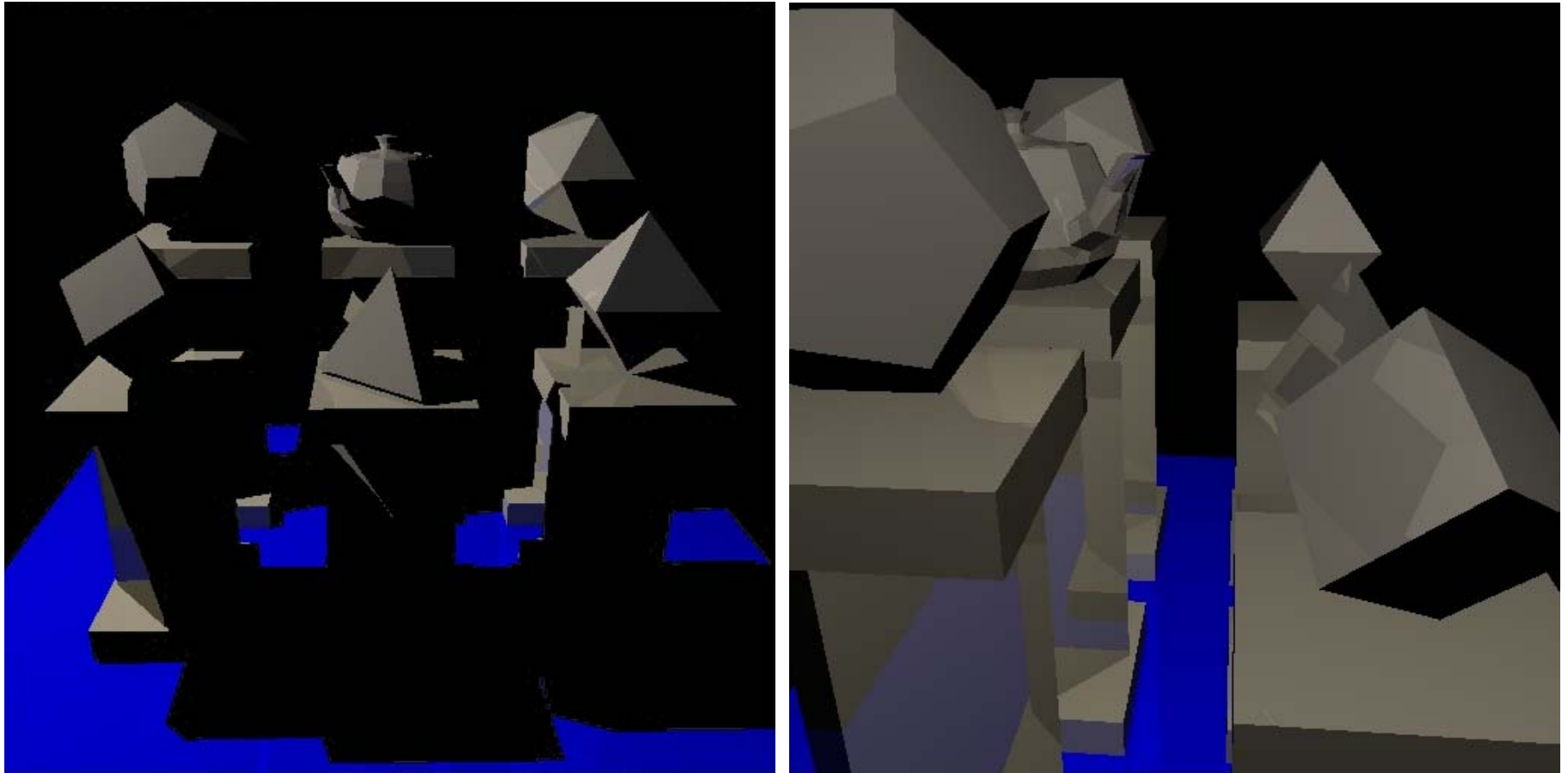
---



Rendered using a Radeon 9700 Pro

# Teapotahedron

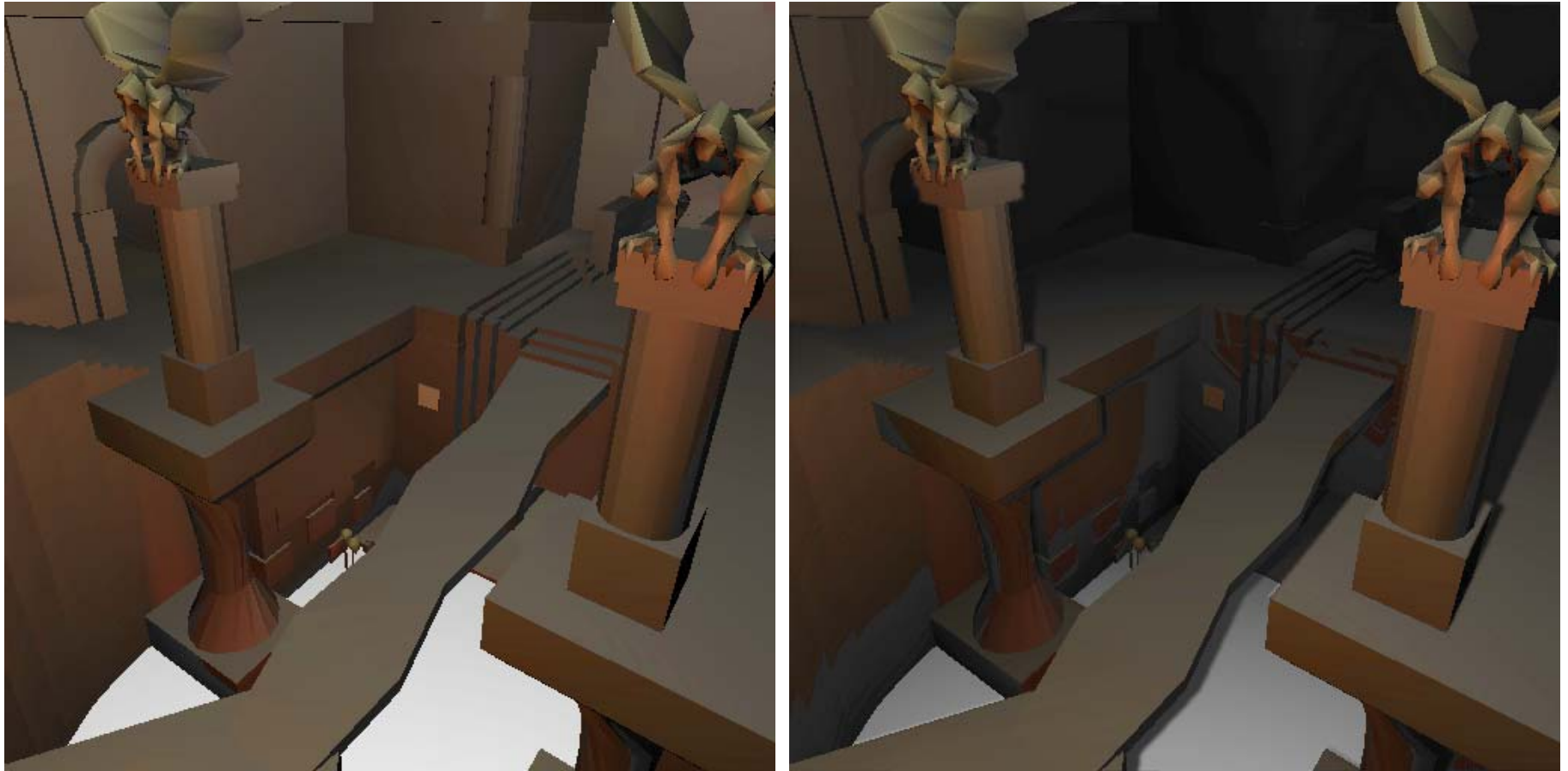
---



Rendered using a Radeon 9700 Pro

# Quake 3 - Ray Traced Shadows

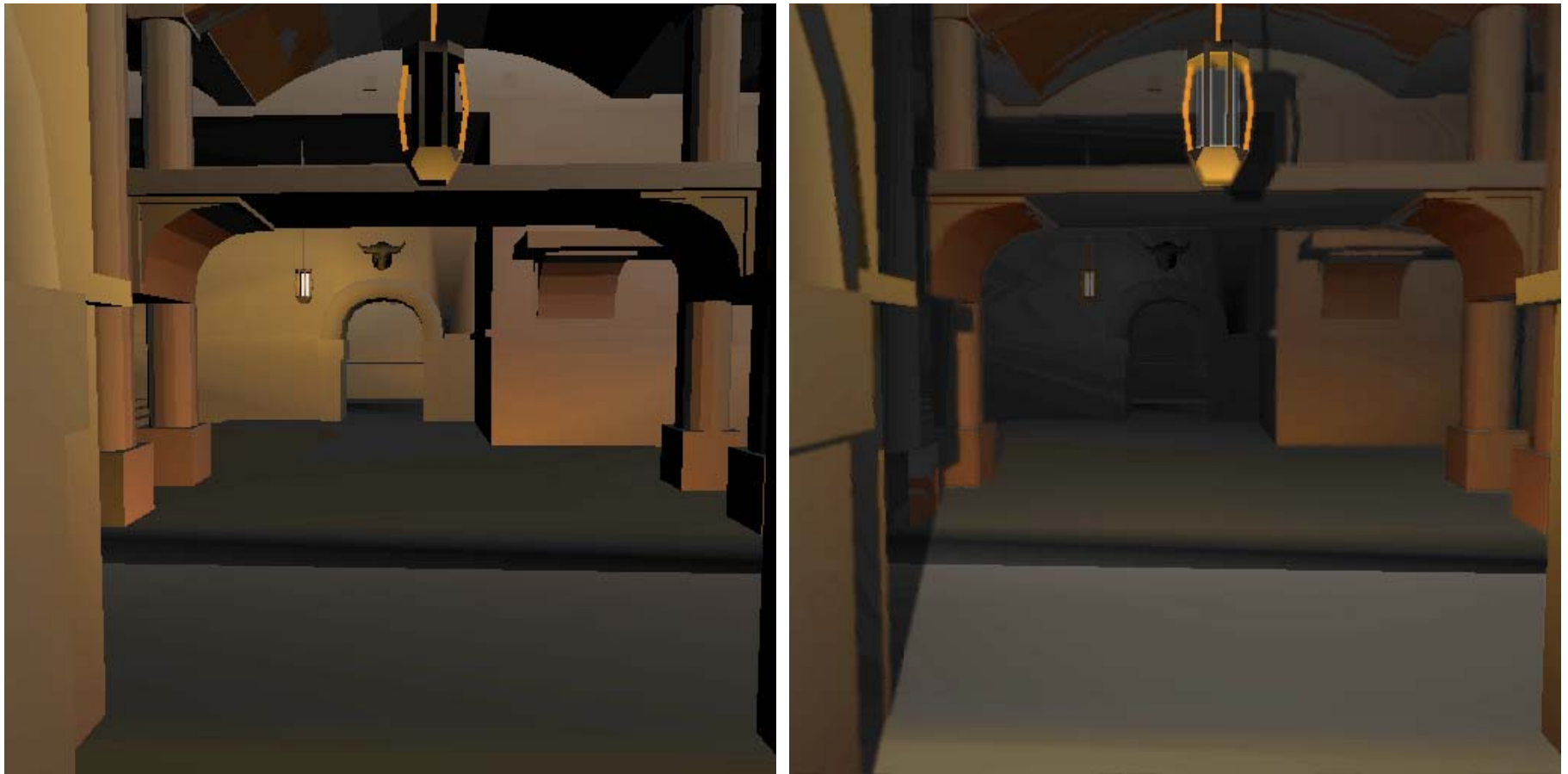
---



Rendered using a Radeon 9700 Pro

# Quake 3 - Ray Traced Shadows

---

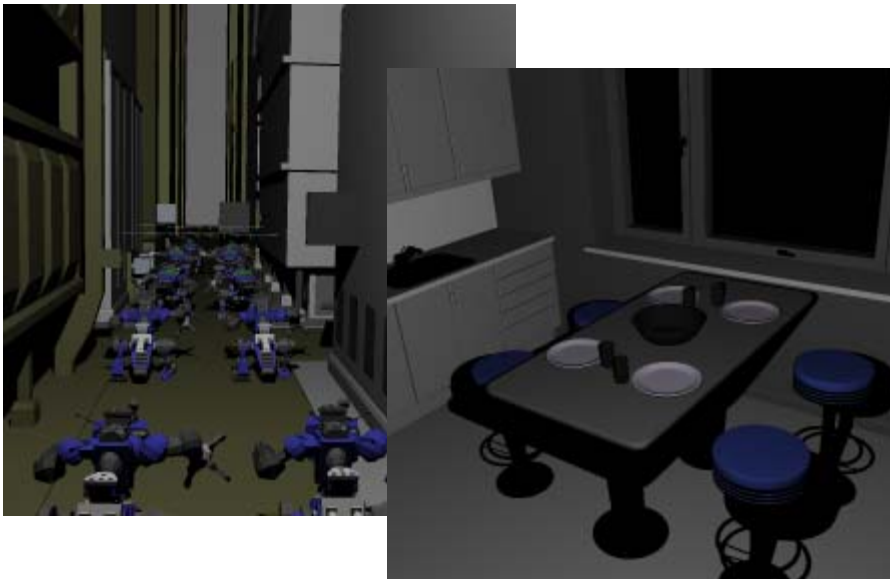


Rendered using a Radeon 9700 Pro

# Recent GPU Ray Tracers

---

- *k*-d tree based acceleration structure



[Foley and Sugerman 2005]

- Bounding volumes

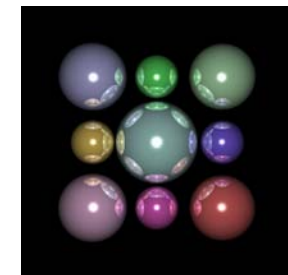
[Thrane and Simonsen 2005]

**GP GPU**

- Grid-based open source ray tracers



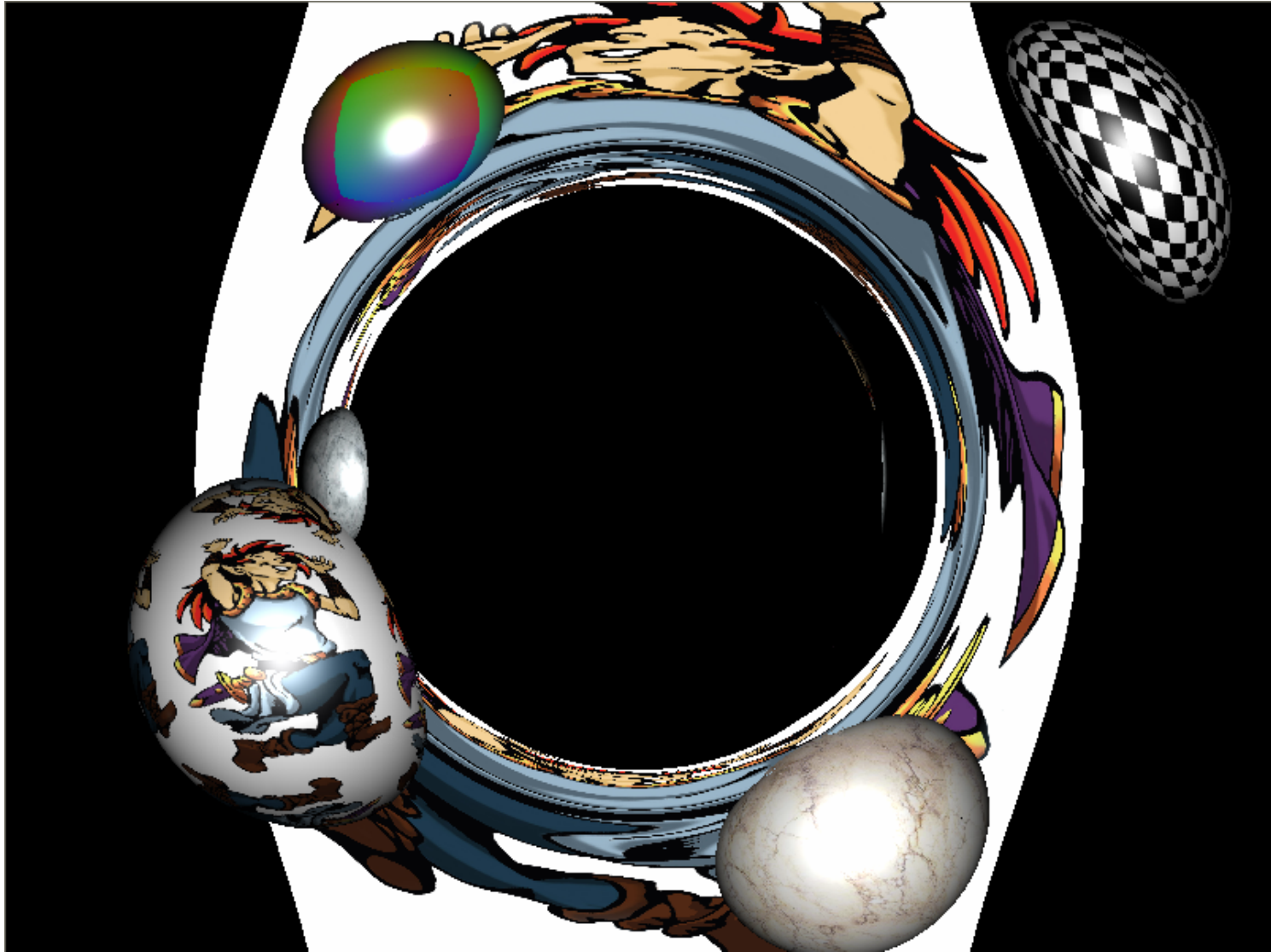
[Christen 2005]



[Karlsson and Ljungstedt 2004]

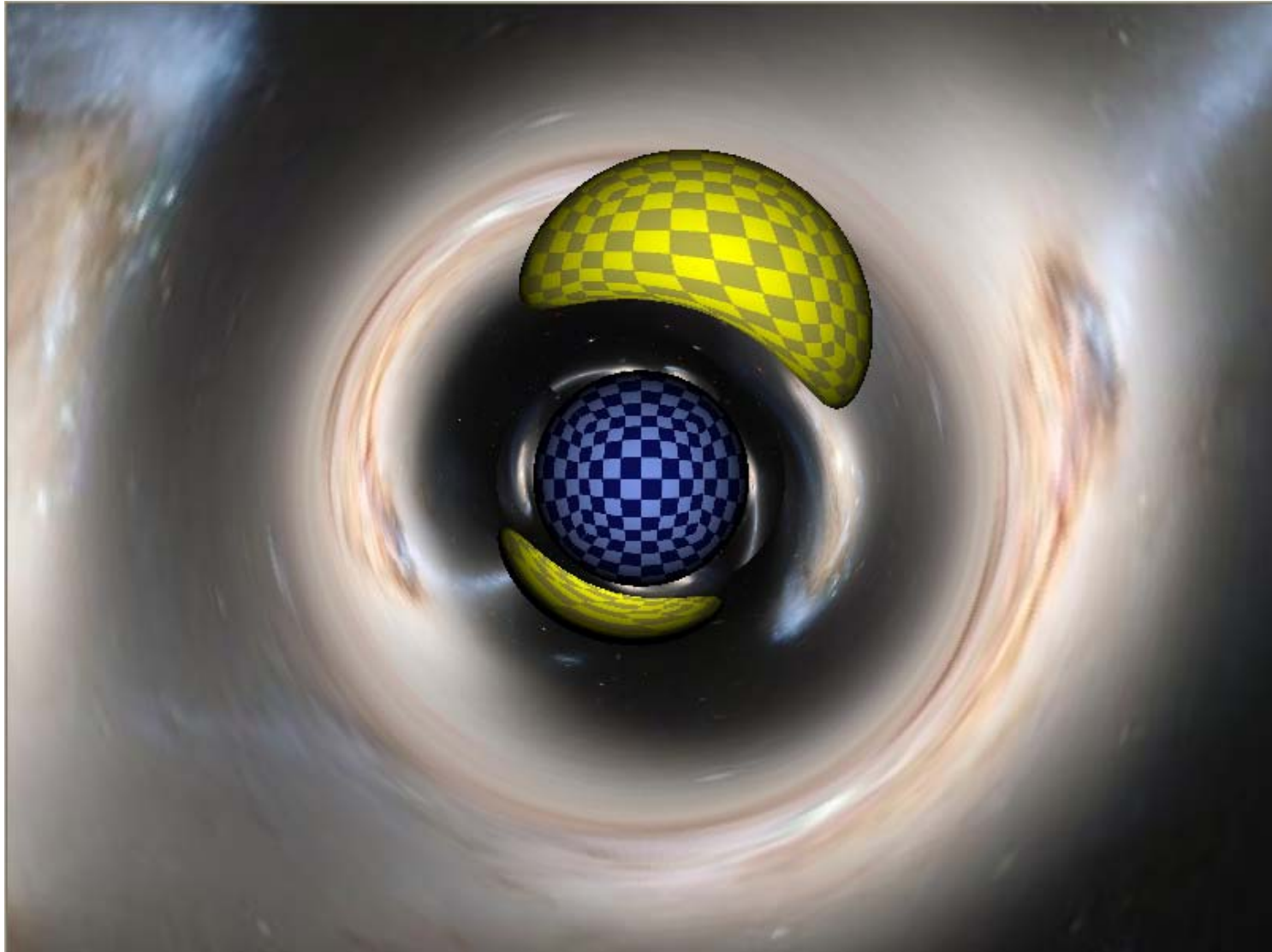
# Nonlinear Ray Tracing

---



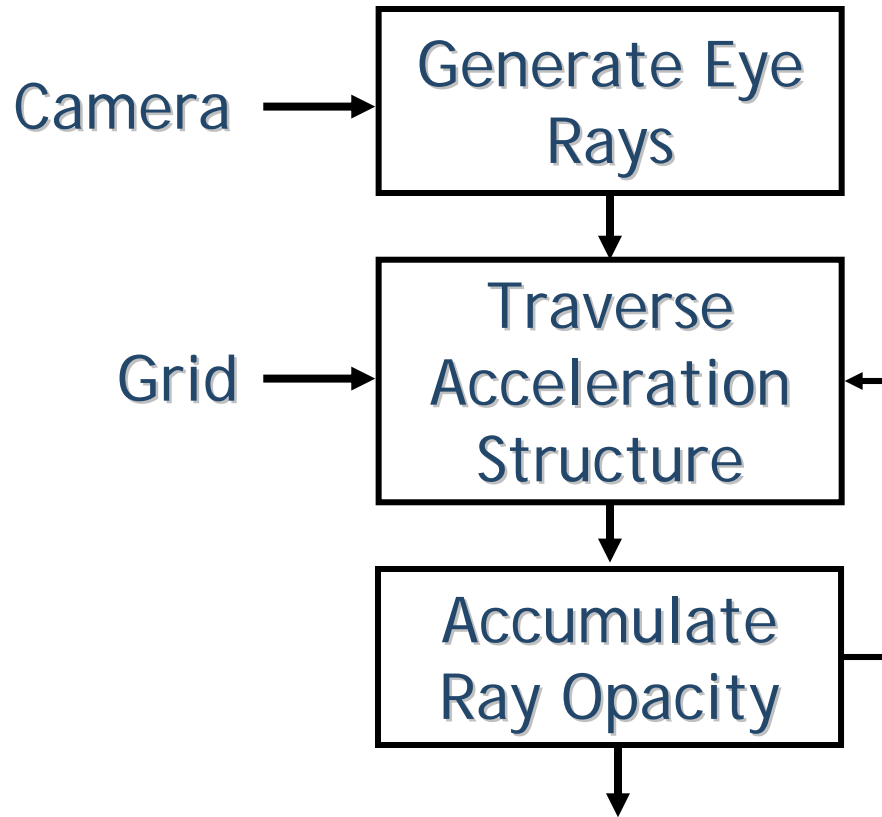
# Nonlinear Ray Tracing

---



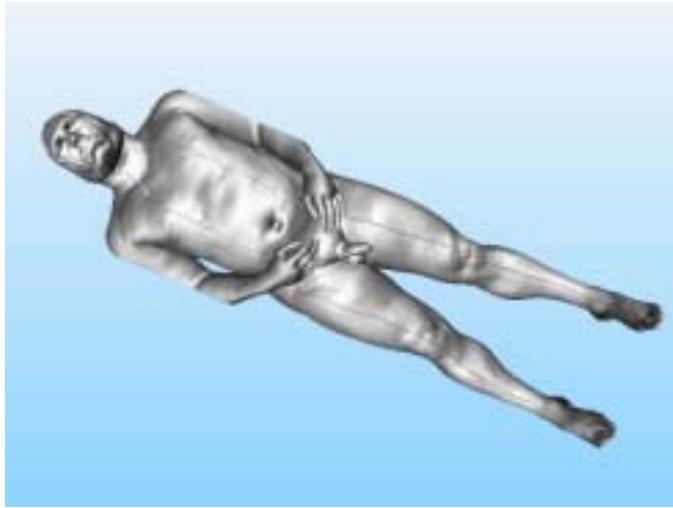
# Simple Volume Ray Caster

---



# Ray Cast Volume Rendering

---



[Krüger and Westermann 2003]



[Roettger et al. 2003]



[Stegmaier et al. 2005]

# Unstructured Meshes

---

- Convex meshes [Weiler et al. 2003]
- Non-convex meshes using depth peeling [Weiler et al. 2005, Bernardon et al. 2004]
- k-buffer [Callahan et al. 2005]
  - not ray casting...

